Kiersten Wener
Marione Ogboi
Team 6

Intro

Frame semantics are highly important to Natural Language Processing techniques as a whole. Being able to identify frames, as described by FrameNet, helps systems to gather a better understanding of the structure and the context of a sentence and/or text. As described by the theory of Lexical Semantics, “the meanings of most words can best be understood on the basis of a semantic frame: a description of a type of event, relation, or entity and the participants in it.” In order for NLP models to take advantage of the rich contextual, structural, and semantic data that FrameNet is able to provide, the model must be able to identify frame elements from within a sentence. A Frame Element is a component of a frame that represents a participant, property or concept that is essential to the meaning described by the frame - it defines the semantic role that different clauses play within that context. For example, the Time Frame Element is a clause that provides time context to the sentence; The Manner Frame Element will describe how an action is being done, etc. If a model were able to identify these frame elements without manual tagging, they would then be able to extract a much richer semantic and structural data that Frame Net provides.

Definitions

From : https://www.nltk.org/howto/framenet.html

- Lexical Unit: “word or phrase that evokes a frame” or a “pairing of a word with meaning”
- Frame: “a script-like conceptual structure that describes a particular type of situation, object, or event along with the participants and props that are needed for that Frame”
- Frame Elements: “The roles in a frame”

Problem Description

FrameNet provides frames, their corresponding Frame Elements, multiple annotated example sentences, and other detailed semantic information for each frame. While this detailed information is useful in understanding sentence meaning, there is currently no widely adopted or fully automated way of extracting Frame Elements from raw text. Frame Elements are crucial to understanding the context and intended meaning of a sentence–they are able to help answer critical questions like *who, what, when, where, why.* This level of semantic understanding is crucial to solve many NLP problems such as question answering, information extraction, event detection, and as well as for furthering developing AIs such as chatbots.

Why are we interested?

If models were able to parse text by a Frame Element, they would be able to give structured meanings to unstructured text. This would allow the system to not only recognize what is being said, but how each individual word relates and contributes to the meaning of the sentence. This would help improve performance in a wide variety of applications like more precise information extraction, question answering, text summarization, and context-aware dialogue systems, helping to bridge the gap between human-level understanding and machine processing of language.

Data

To create our dataset, we used specific annotated examples from FrameNet. We began by analyzing FrameNet and finding the four Frame Elements with the largest annotated sentence sets. These four Frame Elements are Time (8170 sentences), Manner (7612 sentences), Place (7027 sentences) and Degree (7012 sentences). The next largest frame element, Means, had a significantly smaller set of annotated sentences, with only 5045 samples. Because of this, we chose to limit our initial research to just the top four elements.

After finding the Frame Elements with the largest sets of annotated sentences, we were able to begin creating our datasets. Starting with the Time dataset of the Time model, we tokensized each sentence in the sample set and labeled each token according to the B I O Tagging scheme based on the Frame Element annotations given by FrameNet. We used a mapping function to ensure that token labels were properly aligned and implemented masking to ensure each sentence had the same number of tokens.

Using the tokenized sample sentences and their associated labels, we constructed our dataset. The vector of token words was the attribute, and the B I O labels, in numeric form (B : 1, I : 2, O : 0), were the target variable.

We then split the dataset 50/50 into training and validation data. We used a random shuffle split to randomize and divide the data between the two datasets.

Methodology

Final model

To predict which tokens in a sentence belong to a specific Frame Element, we designed a token-level feedforward neural network using token embeddings generated from a pre-trained BERT model.

We decided to use such a simple network because of the straightforward nature of the task - classifying each token in a sentence as B I O (beginning, inside, outside).

We decided to use BERT embedding vectors as input to the network instead of the tokens themselves in order to better capture the context and the relationships between words. This allows the neural network to not only capture the meaning of the sentence, but also the relationships between the words and the role they play in the overall content of the sentence. These embeddings are then passed through a projection layer to adjust the dimensionality to make them more suitable for classification.

Capturing inter-word relationships is crucial for identifying Frame Elements, which represent semantic roles or syntactic functions within a sentence. For example, the Manner Frame Element is used to describe how something is done. This frame element has a direct relationship with the verb in the sentence that it modifies. It is essential that the model is able to identify those patterns.

The classifier head takes the projected BERT embeddings and feeds them through a simple feedforward network composed of:

1. A Linear layer to learn weighted combinations of the input embedding dimensions.
2. A Dropout layer to prevent overfitting by randomly zeroing out a portion of the weights during training.
3. Another Linear layer that outputs logits for each class:
    - **0** → Token is *Outside* any Frame Element
    - **1** → Token is the *Beginning* of a Frame Element
    - **2** → Token is *Inside* a Frame Element

This setup allows us to make a label prediction for each token in the sentence independently, based on its embedding.

During evaluation, we applied postprocessing to eliminate impossible predictions. Based on the BIO labeling technique principles, it is impossible to have a token labeled I directly following a token labeled O. To amend this situation, we decided to experiment with two post processing techniques that take the surrounding labels into consideration. These techniques will be explained in more detail below.

Preprocessing Technique

We started doing our testing using a single Frame Element, Time. After finding a methodology that produced adequate results, we repeated our process on other Frame elements : Degree, Manner, and Place.

First, we searched FrameNet to determine which FrameElements had the most sample sentences. The 5 with the most sample sentences were Time, Manner, Place, Degree, and Means (in descending order).

To create our dataset, we had to begin by extracting all of the sample sentences that included Time as a Frame Element across different Frames. This meant first finding all Frames that had a Frame Element of Time. We then looped through each of those Frames and iterated through each Lexical Unit for that frame. For each Lexical Unit we accessed all of its annotated exemplar sentences and extracted the text, initially labeling all of the character tokens in the text as 0.

This is the beginning of our BIO labeling for each sentence. We extract the character spans for each Time Frame Element labeling the first character as B-Time (beginning of the Time frame Element) and the subsequent character with I-Time (inside the Frame Element). All characters that aren't part of the Frame element are labeled as 0, which represents O in BIO labeling, meaning it is outside of the Frame Element.

We then use offset mapping and our custom align_labels_with_tokens() function to map the character level BIO labeling to the BERT tokens. Finally, we turn the labels numeric and convert the input IDs, attention masks, and labels to pytorch tensors. These tensors will then be turned into our dataset, which is later split into training and validation datasets.

Post Processing Techniques

We experimented with multiple post-processing techniques, each of which takes a different degree of surrounding tokens into consideration when deciding how to fix an impossible label output sequence. After much trial and error with the results from the Time Frame Element model, we selected two techniques to implement and compare results with on the remaining FEs.

The first post processing technique (the function fix_bio_predictions_2()) was the simpler of the two processes. If checks to see if there is a "0 2" invalid output label sequence. If so, the function considers the label following the "invalid" 2. If the next label is also a 2, then the 0 label of the invalid sequence is forced to a 1 or a 2 (depending on the token before it).

In the second post processing technique (the function fix_bio_predictions3()), If an I tag (2) follows an O tag (0) and is part of a longer run of I tags (O I I), then the preceding O is changed to a B.

Results

Across Frame Elements

To evaluate our models, we are using a metric based on how many full Frame Elements are correctly identified, meaning every token in the Frame Element is correctly labeled. This metric is much more applicable than accuracy, precision, or recall because a Frame Element is a single unit, and if one of the tokens in the unit is not labeled correctly, the sequence is meaningless in many practical applications. For example, if a model mislabels 1 token in every Frame element, it would still result in a high accuracy even though the model is performing very poorly.

- The strict match accuracy = (number of Frame Elements with all tokens correctly identified)/(Total number of Frame Element Sequences)
- Partial match accuracy = (number of Frame Elements where at least some tokens are correctly identified)/(Total number of Frame Element Sequences)

Frame Element Name	Strict match accuracy	Partial match accuracy
Time	0.841	0.973
Degree	0.816	0.902
Manner	0 .797	0.913
Place	0.734	0.885

As shown in the table above, our model produced promising results across four frame elements. The highest strict match accuracy was achieved for **Time** (0.841), followed closely by **Degree** (0.816). **Place** and **Manner** showed slightly lower performance, with strict accuracies of 0.734 and 0.797 respectively. However, the partial match accuracies for all Frame Elements remained above 0.88, with **Time** achieving an especially high partial accuracy of 0.973, suggesting the model frequently captures at least some relevant portion of the span

Passing in random sentences:

I passed in a variety of “random”sentences of different lengths, structures, and subjects to each of the models. Each example sentence was designed to test the limits and strengths of the models.

For example, for the Place Frame Element model, I passed in multiple sentences where the Frame Element was a long clause like “We met the boy in the blue shirt at the lagoon across the street after dark” as well as short, simple sentences like “I want to go home”.

Analysis

I found that the models perform well on short, descriptive phrases associated with Manner and Degree FEs. They had a worse performance when the Frame Element was a longer phrase. For example, the Manner FE model was able to consistently correctly identify the FEs “quickly” and “softly”.However, performance dropped when the Frame Element was a longer or more syntactically complex phrase. In one example, for the Manner FE “with a rag”, the model only labeled the preposition “with”, failing to capture the rest of the phrase.

From my analysis, the models with the best results were for Frame Elements Time and Degree because of their simple and consistent structure across sentences. You can be fairly certain that the Degree FE in a sentence will be an adverb or a short descriptive phrase less than 3 words (slightly, barely). Time FEs are also structured consistently across sentences and contexts. They also all include some kind of identifying temporal phrase, like “noon” or “yesterday”. These temporal words are used uniquely to describe Time, a pattern that is simple for the network to recognize, leading to the Time model’s high performance.

However, the Frame Elements Manner and Place have more variance in sentence structure across usages. A Manner FE can be a simple adverb, a prepositional phrase, or phrase that denotes means, like “by hand”. Place can be expressed through prepositional phrases, adverbials (here, there), or a noun phrase. It is also more common for Manner and Place FEs to contain ambiguous words. When looking at the phrase “In the park”, “park” can have different meanings in different contexts, making it more complicated for the model to identify it as a Place FE. The same concept applies to Manner. The phrase “in a rush” could apply to a Manner FE, or it could potentially apply to a Place FE (ex “He stood under the rush of water”).

A second factor that I believe plays a role in the lesser performance of the models for Manner and Place is scarcity. With the Time and Degree Frame Elements, the real life vocabulary is much more limited than the vocabularies for Manner and Place. There are a finite number of ways to describe time: a finite number of hours, days, months, etc. As for Degree, there is a finite number of modifiers that can be a Degree FE: too, a lot. This limited vocabulary makes it much easier for the training data to accurately represent the total real life vocabulary, meaning the models will have better performance when predicting on unseen data.

On the other hand, there is an almost unlimited number of ways to describe a Place or a Manner. The real life vocabulary for these Frame Elements is almost unbounded due to the unlimited combinations of prepositions and nouns, like “With a rag”, “with a mop”, “with a spoon” and basically any other noun at the end of that phrase is a possible manner for Frame Element. The same concept can be applied to the Place FEs. This issue makes it much more difficult for the real life uses of the Manner and Place Frame Elements to be represented fully in the training

data. There is a large combination of words and phrases making up these FEs with a structure that is unseen by the models, making it more difficult for the models to accurately label the Place and Manner FEs.

I have also found that different Frame Element models produced better results using different post processing techniques. It may be worth looking at how part of speech and Frame Element structure affect predictions/how incorrect labels should be handled

What's next

- Experiment with more complex network structures
    - Incorporate a backward pass to see if it helps to better capture the relationships between words
    - Find a model that is able to learn the legal output label sequences instead of having to adjust them with post processing
- Different post processing techniques
    - Experiment with the number of surrounding tokens taken into consideration when trying to fix an invalid output sequence
- Test different FE models